

Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-02/6755.html>

From: George Anzinger (george_at_mvista.com)

Date: 02/26/04

Date: Wed, 25 Feb 2004 15:10:03 -0800
To: Tom Rini <trini@kernel.crashing.org>

Convention has been that control C does the break. If I read this correctly you are saying that any character does it. If that is the intent and it works, then the whole buffer interrupt characters thing can be removed. In fact, an interrupt implies we are not in kgdb (it holds all interrupts off) so that test, too, can go. The interrupt thing then reduces to a breakpoint.

-g

Tom Rini wrote:

```
> The following updates the serial driver with the fixes / cleanups that
> are in George's version of the driver. There's a few slightly 'odd'
> things in this patch, which stem from the fact that in my next round of
> patches there will (a) be kernel/Kconfig.kgdb and (b) 1 kgdb i/o driver
> at a time.
>
> # This is a BitKeeper generated patch for the following project:
> # Project Name: Linux kernel tree
> # This patch format is intended for GNU patch command version 2.5 or higher.
> # This patch includes the following deltas:
> # ChangeSet 1.1662 -> 1.1663
> # drivers/serial/kgdb_8250.c 1.2 -> 1.3
> # kernel/kgdb.c 1.2 -> 1.3
> # drivers/serial/Kconfig 1.21 -> 1.22
> # include/linux/kgdb.h 1.2 -> 1.3
> #
> # The following is the BitKeeper ChangeSet Log
> # -----
> # 04/02/25 trini@kernel.crashing.org 1.1663
> # Serial update.
> # -----
> #
> diff -Nru a/drivers/serial/Kconfig b/drivers/serial/Kconfig
> --- a/drivers/serial/Kconfig Wed Feb 25 14:21:26 2004
> +++ b/drivers/serial/Kconfig Wed Feb 25 14:21:26 2004
> @@ -13,6 +13,27 @@
```

Linux-Kernel: Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

```
> Uses generic serial port (8250) for kgdb. This is independent of the
> option 9250/16550 and compatible serial port.
>
> +config KGDB_PORT
> + hex "hex I/O port address of the debug serial port"
> + depends on KGDB_8250
> + default 3f8
> + help
> + Some systems (x86 family at this writing) allow the port
> + address to be configured. The number entered is assumed to be
> + hex, don't put 0x in front of it. The standard address are:
> + COM1 3f8, irq 4 and COM2 2f8 irq 3. Setserial /dev/ttySx
> + will tell you what you have. It is good to test the serial
> + connection with a live system before trying to debug.
> +
> +config KGDB_IRQ
> + int "IRQ of the debug serial port"
> + depends on KGDB_8250
> + default 4
> + help
> + This is the irq for the debug port. If everything is working
> + correctly and the kernel has interrupts on a control C to the
> + port should cause a break into the kernel debug stub.
> +
> #
> # The new 8250/16550 serial drivers
> config SERIAL_8250
> diff -Nru a/drivers/serial/kgdb_8250.c b/drivers/serial/kgdb_8250.c
> --- a/drivers/serial/kgdb_8250.c Wed Feb 25 14:21:26 2004
> +++ b/drivers/serial/kgdb_8250.c Wed Feb 25 14:21:26 2004
> @@ -1,137 +1,218 @@
> /*
> * 8250 interface for kgdb.
> *
> - * Restructured for making kgdb capable of handling different types of serial
> - * interfaces, by Amit Kale (amitkale@emsyssoft.com)
> - *
> - * Written (hacked together) by David Grothe (dave@gcom.com)
> - *
> - * Modified by Scott Foehner (sfoehner@engr.sgi.com) to allow connect
> - * on boot-up
> + * This is a merging of many different drivers, and all of the people have
> + * had an impact in some form or another:
> *
> + * Amit Kale <amitkale@emsyssoft.com>
> + * David Grothe <dave@gcom.com>
> + * Scott Foehner <sfoehner@engr.sgi.com>
> + * George Anzinger <george@mvista.com>
> + * Robert Walsh <rjwalsh@durables.org>
> + * wangdi <>wangdi@clusterfs.com>
> + * San Mehat
```

```
> + * Tom Rini <trini@mvista.com>
> */
>
> -#include <linux/module.h>
> -#include <linux/errno.h>
> -#include <linux/signal.h>
> -#include <linux/sched.h>
> -#include <linux/timer.h>
> +#include <linux/config.h>
> +#include <linux/kernel.h>
> +#include <linux/init.h>
> +#include <linux/spinlock.h>
> +#include <linux/kgdb.h>
> #include <linux/interrupt.h>
> #include <linux/tty.h>
> -#include <linux/tty_flip.h>
> #include <linux/serial.h>
> #include <linux/serial_core.h>
> #include <linux/serial_reg.h>
> #include <linux/serialP.h>
> -#include <linux/config.h>
> -#include <linux/major.h>
> -#include <linux/string.h>
> -#include <linux/fcntl.h>
> -#include <linux/termios.h>
> -#include <linux/kgdb.h>
>
> -#include <asm/system.h>
> #include <asm/io.h>
> -#include <asm/segment.h>
> -#include <asm/bitops.h>
> -#include <asm/system.h>
> -#include <asm/irq.h>
> -#include <asm/atomic.h>
> +#include <asm/serial.h> /* For BASE_BAUD and SERIAL_PORT_DFNS */
>
> #define GDB_BUF_SIZE 512 /* power of 2, please */
>
> +#if defined(CONFIG_KGDB_9600BAUD)
> +#define SERIAL_BAUD 9600
> +#elif defined(CONFIG_KGDB_19200BAUD)
> +#define SERIAL_BAUD 19200
> +#elif defined(CONFIG_KGDB_38400BAUD)
> +#define SERIAL_BAUD 38400
> +#elif defined(CONFIG_KGDB_57600BAUD)
> +#define SERIAL_BAUD 57600
> +#elif defined(CONFIG_KGDB_115200BAUD)
> +#define SERIAL_BAUD 115200
> +#else
> +#define SERIAL_BAUD 115200 /* Start with this if not given */
> +#endif
```

```

> +
> +#if defined(CONFIG_KGDB_TTYS0)
> +#define KGDB_PORT 0
> +#elif defined(CONFIG_KGDB_TTYS1)
> +#define KGDB_PORT 1
> +#elif defined(CONFIG_KGDB_TTYS2)
> +#define KGDB_PORT 2
> +#elif defined(CONFIG_KGDB_TTYS3)
> +#define KGDB_PORT 3
> +#else
> +#define KGDB_PORT 0 /* Start with this if not given */
> +#endif
> +
> +int kgdb8250_baud = SERIAL_BAUD;
> +int kgdb8250_ttyS = KGDB_PORT;
> +
> +static char kgdb8250_buf[GDB_BUF_SIZE];
> +static int kgdb8250_buf_in_inx;
> +static atomic_t kgdb8250_buf_in_cnt;
> +static int kgdb8250_buf_out_inx;
> +
> -static int kgdb8250_got_dollar = -3, kgdb8250_got_H = -3,
> -kgdb8250_interrupt_iteration = 0;
> +/* Determine serial information. */
> +static struct serial_state state = {
> +.magic = SSTATE_MAGIC,
> +.baud_base = BASE_BAUD,
> +.custom_divisor = BASE_BAUD / SERIAL_BAUD,
> +};
> +
> -/* We only allow for 4 ports to be registered. We default to standard
> -* PC values. */
> -static struct uart_port rs_table[4] = {
> -{.line = 0x3f8,.irq = 4},
> -{.line = 0x2f8,.irq = 3},
> -{.line = 0x3e8,.irq = 4},
> -{.line = 0x2e8,.irq = 3},
> +static struct async_struct gdb_async_info = {
> +.magic = SERIAL_MAGIC,
> +.state = &state,
> +.tty = (struct tty_struct *) &state,
> +};
> -static void (*serial_outb) (unsigned char, unsigned long);
> -static unsigned long (*serial_inb) (unsigned long);
> +
> +/* Space between registers. */
> +static int reg_shift;
> +
> +/* Not all arches define this. */
> +#ifndef SERIAL_PORT_DFNS
> +#define SERIAL_PORT_DFNS

```

```
> +#endif
> +static struct serial_state rs_table[] = {
> + SERIAL_PORT_DFNS /* defined in <asm/serial.h> */
> +};
> +
> +/* Do we need to look in the rs_table? */
> +static int serial_from_rs_table = 0;
> +
> +static void (*serial_outb) (unsigned char val, unsigned long addr);
> +static unsigned long (*serial_inb) (unsigned long addr);
>
> int serial8250_release_irq(int irq);
>
> -int kgdb8250_irq;
> -unsigned long kgdb8250_port;
> +static int kgdb8250_init(void);
> +static unsigned long kgdb8250_port;
>
> -int kgdb8250_baud = 115200;
> -int kgdb8250_ttyS;
> +static int initialized = -1;
>
> -static unsigned long direct_inb(unsigned long addr)
> +static unsigned long
> +direct_inb(unsigned long addr)
> {
> return readb(addr);
> }
>
> -static void direct_outb(unsigned char val, unsigned long addr)
> +static void
> +direct_outb(unsigned char val, unsigned long addr)
> {
> writeb(val, addr);
> }
>
> -static unsigned long io_inb(unsigned long port)
> +static unsigned long
> +io_inb(unsigned long port)
> {
> return inb(port);
> }
>
> -static void io_outb(unsigned char val, unsigned long port)
> +static void
> +io_outb(unsigned char val, unsigned long port)
> {
> outb(val, port);
> }
>
> /*
```

Linux-Kernel: Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

```
> - * Get a byte from the hardware data buffer and return it
> - * Get a char if available, return -EAGAIN if nothing available.
> + * Wait until the interface can accept a char, then write it.
> */
> -static int read_data_bfr(void)
> +void
> +kgdb_put_debug_char(int chr)
> {
> - if (serial_inb(kgdb8250_port + UART_LSR) & UART_LSR_DR)
> - return (serial_inb(kgdb8250_port + UART_RX));
> + while (!(serial_inb(kgdb8250_port + (UART_LSR << reg_shift)) &
> + UART_LSR_THRE))
> + ;
>
> - return -EAGAIN;
> + serial_outb(chr, kgdb8250_port + (UART_TX << reg_shift));
> }
>
> /*
> - * Empty the receive buffer first, then look at the interface hardware.
> - * It waits for a character from the serial interface and then returns it.
> + * Get a byte from the hardware data buffer and return it
> */
> -static int kgdb8250_read_char(void)
> +static int
> +read_data_bfr(void)
> {
> - int retchar;
> - if (atomic_read(&kgdb8250_buf_in_cnt) != 0) {
> - /* intr routine has q'd chars read them from buffer */
> - int chr;
> + char it = serial_inb(kgdb8250_port + (UART_LSR << reg_shift));
>
> - chr = kgdb8250_buf[kgdb8250_buf_out_inx++];
> - kgdb8250_buf_out_inx &= (GDB_BUF_SIZE - 1);
> - atomic_dec(&kgdb8250_buf_in_cnt);
> - return chr;
> + if (it & UART_LSR_DR)
> + return serial_inb(kgdb8250_port + (UART_RX << reg_shift));
> +
> + /*
> + * If we have a framing error assume somebody messed with
> + * our uart. Reprogram it and send '-' both ways...
> + */
> + if (it & 0xc) {
> + kgdb8250_init();
> + kgdb_put_debug_char('-');
> + return '-';
> }
> - do {
> - /* read from hardware */
```

```

> - retchar = read_data_bfr();
> - } while (retchar < 0);
> - return retchar;
> +
> + return -1;
> }
>
> /*
> - * Wait until the interface can accept a char, then write it.
> + * Get a char if available, return -1 if nothing available.
> + * Empty the receive buffer first, then look at the interface hardware.
> + *
> + * Locking here is a bit of a problem. We MUST not lock out communication
> + * if we are trying to talk to gdb about a kgdb entry. ON the other hand
> + * we can loose chars in the console pass thru if we don't lock. It is also
> + * possible that we could hold the lock or be waiting for it when kgdb
> + * NEEDS to talk. Since kgdb locks down the world, it does not need locks.
> + * We do, of course have possible issues with interrupting a uart operation,
> + * but we will just depend on the uart status to help keep that straight.
> */
> -static void kgdb8250_write_char(int chr)
> +
> +static spinlock_t uart_interrupt_lock = SPIN_LOCK_UNLOCKED;
> +#ifdef CONFIG_SMP
> +extern spinlock_t kgdb_spinlock;
> +#endif
> +
> +int
> +kgdb_get_debug_char(void)
> {
> - while (!(serial_inb(kgdb8250_port + UART_LSR) & UART_LSR_THRE))
> - /* Do nothing */;
> + int retchr;
> + unsigned long flags;
> + local_irq_save(flags);
> +#ifdef CONFIG_SMP
> + if (!spin_is_locked(&kgdb_spinlock)) {
> + spin_lock(&uart_interrupt_lock);
> + }
> +#endif
> + /* intr routine has q'd chars */
> + if (atomic_read(&kgdb8250_buf_in_cnt) != 0) {
> + retchr = kgdb8250_buf[kgdb8250_buf_out_inx++];
> + kgdb8250_buf_out_inx &= (GDB_BUF_SIZE - 1);
> + atomic_dec(&kgdb8250_buf_in_cnt);
> + goto out;
> + }
> +
> + do {
> + retchr = read_data_bfr();
> + } while (retchr < 0);

```

```

>
> - serial_outb(chr, kgdb8250_port + UART_TX);
> +out:
> +#ifdef CONFIG_SMP
> + if (!spin_is_locked(&kgdb_spinlock)) {
> + spin_unlock(&uart_interrupt_lock);
> + }
> +#endif
> + local_irq_restore(flags);
>
> + return retchr;
> }
>
> /*
> @@ -139,12 +220,12 @@
> * It will receive a limited number of characters of input
> * from the gdb host machine and save them up in a buffer.
> *
> - * When kgdb8250_read_char() is called it
> + * When kgdb_get_debug_char() is called it
> * draws characters out of the buffer until it is empty and
> * then reads directly from the serial port.
> *
> * We do not attempt to write chars from the interrupt routine
> - * since the stubs do all of that via kgdb8250_write_char() which
> + * since the stubs do all of that via kgdb_put_debug_char() which
> * writes one byte after waiting for the interface to become
> * ready.
> *
> @@ -156,15 +237,27 @@
> * care to learn can make this work for any low level serial
> * driver.
> */
> -static irqreturn_t kgdb8250_interrupt(int irq, void *dev_id,
> - struct pt_regs *regs)
> +static irqreturn_t
> +kgdb8250_interrupt(int irq, void *dev_id, struct pt_regs *regs)
> {
> - int chr;
> - int iir;
> + int chr, iir;
> + unsigned long flags;
> +
> + if (irq != gdb_async_info.line)
> + return IRQ_NONE;
> +
> + /* If we get an interrupt, then KGDB is trying to connect. */
> + if (!kgdb_connected) {
> + breakpoint();
> + return IRQ_HANDLED;
> + }

```

```

> +
> + local_irq_save(flags);
> + spin_lock(&uart_interrupt_lock);
>
> do {
> chr = read_data_bfr();
> - iir = serial_inb(kgdb8250_port + UART_IIR);
> + iir = serial_inb(kgdb8250_port + (UART_IIR << reg_shift));
> if (chr < 0)
> continue;
>
> @@ -174,28 +267,6 @@
> continue;
> }
>
> - if (atomic_read(&kgdb_killed_or_detached)) {
> - if (chr == '$')
> - kgdb8250_got_dollar =
> - kgdb8250_interrupt_iteration;
> - if (kgdb8250_interrupt_iteration ==
> - kgdb8250_got_dollar + 1 && chr == 'H')
> - kgdb8250_got_H = kgdb8250_interrupt_iteration;
> - else if (kgdb8250_interrupt_iteration ==
> - kgdb8250_got_H + 1 && chr == 'c') {
> - kgdb8250_buf[kgdb8250_buf_in_inx++] = chr;
> - atomic_inc(&kgdb8250_buf_in_cnt);
> - atomic_set(&kgdb_might_be_resumed, 1);
> - wmb();
> - breakpoint();
> - atomic_set(&kgdb_might_be_resumed, 0);
> - kgdb8250_interrupt_iteration = 0;
> - kgdb8250_got_dollar = -3;
> - kgdb8250_got_H = -3;
> - continue;
> - }
> - }
> -
> if (atomic_read(&kgdb8250_buf_in_cnt) >= GDB_BUF_SIZE) {
> /* buffer overflow, clear it */
> kgdb8250_buf_in_inx = 0;
> @@ -207,33 +278,30 @@
> kgdb8250_buf[kgdb8250_buf_in_inx++] = chr;
> kgdb8250_buf_in_inx &= (GDB_BUF_SIZE - 1);
> atomic_inc(&kgdb8250_buf_in_cnt);
> - }
> - while (iir & UART_IIR_RDI);
> + } while (iir & UART_IIR_RDI);
>
> - if (atomic_read(&kgdb_killed_or_detached))
> - kgdb8250_interrupt_iteration++;
> + spin_unlock(&uart_interrupt_lock);

```

```

> + local_irq_restore(flags);
>
> return IRQ_HANDLED;
> -
> }
>
> /*
> - * Initializes serial port.
> - * ttyS - integer specifying which serial port to use for debugging
> - * baud - baud rate of specified serial port
> + * Returns:
> + * 0 on success, 1 on failure.
> */
> -static int kgdb8250_init(int ttyS, int baud)
> +static int
> +kgdb8250_init(void)
> {
> unsigned cval;
> int bits = 8;
> int parity = 'n';
> int cflag = CREAD | HUPCL | CLOCAL;
> - int quot = 0;
>
> /*
> * Now construct a cflag setting.
> */
> - switch (baud) {
> + switch (kgdb8250_baud) {
> case 1200:
> cflag |= B1200;
> break;
> @@ -256,7 +324,7 @@
> cflag |= B115200;
> break;
> default:
> - baud = 9600;
> + kgdb8250_baud = 9600;
> /* Fall through */
> case 9600:
> cflag |= B9600;
> @@ -287,7 +355,6 @@
> *
> */
>
> - quot = (1843200 / 16) / baud;
> cval = cflag & (CSIZE | CSTOPB);
> cval >>= 4;
> if (cflag & PARENB)
> @@ -300,75 +367,118 @@
> * and set speed.
> */

```

```

> cval = 0x3;
> - serial_outb(cval | UART_LCR_DLAB, kgdb8250_port + UART_LCR); /* set DLAB */
> - serial_outb(quot & 0xff, kgdb8250_port + UART_DLL); /* LS of divisor */
> - serial_outb(quot >> 8, kgdb8250_port + UART_DLM); /* MS of divisor */
> - serial_outb(cval, kgdb8250_port + UART_LCR); /* reset DLAB */
> - serial_outb(UART_IER_RDI, kgdb8250_port + UART_IER); /* turn on interrupts */
> + /* set DLAB */
> + serial_outb(cval | UART_LCR_DLAB, kgdb8250_port +
> + (UART_LCR << reg_shift));
> + /* LS */
> + serial_outb(gdb_async_info.state->custom_divisor & 0xff,
> + kgdb8250_port + (UART_DLL << reg_shift));
> + /* MS */
> + serial_outb(gdb_async_info.state->custom_divisor >> 8,
> + kgdb8250_port + (UART_DLM << reg_shift));
> + /* reset DLAB */
> + serial_outb(cval, kgdb8250_port + (UART_LCR << reg_shift));
> + /* turn on interrupts */
> + serial_outb(UART_IER_RDI, kgdb8250_port + (UART_IER << reg_shift));
> serial_outb(UART_MCR_OUT2 | UART_MCR_DTR | UART_MCR_RTS,
> - kgdb8250_port + UART_MCR);
> + kgdb8250_port + (UART_MCR << reg_shift));
>
> /*
> * If we read 0xff from the LSR, there is no UART here.
> */
> - if (serial_inb(kgdb8250_port + UART_LSR) == 0xff)
> - return -ENODEV;
> + if (serial_inb(kgdb8250_port + (UART_LSR << reg_shift)) == 0xff)
> + return -1;
> return 0;
> }
>
> -int kgdb8250_hook(void)
> +int
> +kgdb_hook_io(void)
> {
> int retval;
>
> - /*
> - * Set port and irq number.
> - */
> - kgdb8250_irq = rs_table[kgdb8250_ttyS].irq;
> - switch (rs_table[kgdb8250_ttyS].iotype) {
> + /* Setup any serial port information we may need to */
> + #ifdef CONFIG_KGDB_SIMPLE_SERIAL
> + /* We must look in the rs_table[]. */
> + serial_from_rs_table = 1;
> + #endif
> + /* If the user has overridden our definitions, or if we've only
> + * been told the ttyS to use, look at rs_table. */

```

Linux-Kernel: Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

```
> + if (serial_from_rs_table) {
> + /* The user has selected one of ttyS[0-3], which we pull
> + * from rs_table[.]. If this doesn't exist, user error. */
> + gdb_async_info.port = gdb_async_info.state->port =
> + rs_table[KGDB_PORT].port;
> + gdb_async_info.line = gdb_async_info.state->irq =
> + rs_table[KGDB_PORT].irq;
> + gdb_async_info.state->io_type = rs_table[KGDB_PORT].io_type;
> + reg_shift = rs_table[KGDB_PORT].iomem_reg_shift;
> + }
> +
> + switch (gdb_async_info.state->io_type) {
> case SERIAL_IO_MEM:
> - kgdb8250_port = (unsigned long)rs_table[kgdb8250_ttyS].membase;
> - serial_inb = direct_inb;
> + kgdb8250_port = (unsigned long)
> + rs_table[kgdb8250_ttyS].iomem_base;
> serial_outb = direct_outb;
> + serial_inb = direct_inb;
> break;
> + case SERIAL_IO_PORT:
> default:
> - kgdb8250_port = rs_table[kgdb8250_ttyS].line;
> - serial_inb = io_inb;
> + kgdb8250_port = rs_table[kgdb8250_ttyS].port;
> serial_outb = io_outb;
> + serial_inb = io_inb;
> }
>
> + #ifndef CONFIG_KGDB_SIMPLE_SERIAL
> + /* The user has provided the IRQ and I/O location. */
> + kgdb8250_port = gdb_async_info.port = gdb_async_info.state->port =
> + CONFIG_KGDB_PORT;
> + gdb_async_info.line = gdb_async_info.state->irq = CONFIG_KGDB_IRQ;
> + #endif
> +
> + #ifdef CONFIG_SERIAL_8250
> - if ((retval = serial8250_release_irq(kgdb8250_irq)) < 0)
> - return retval;
> + if (serial8250_release_irq(gdb_async_info.line))
> + return -1;
> + #endif
>
> - if ((retval = kgdb8250_init(kgdb8250_ttyS, kgdb8250_baud)) < 0)
> - return retval;
> + if (kgdb8250_init() == -1)
> + return -1;
>
> - retval = request_irq(kgdb8250_irq, kgdb8250_interrupt, SA_INTERRUPT,
> - "GDB-stub", NULL);
> - return retval;
```

```

> -}
> + retval = request_irq(gdb_async_info.line, kgdb8250_interrupt,
> + SA_INTERRUPT, "GDB-stub", NULL);
> + if (retval == 0)
> + initialized = 1;
> + else
> + initialized = 0;
>
> -void kgdb8250_add_port(int i, struct uart_port *serial_req)
> -{
> - rs_table[i].iotype = serial_req->iotype;
> - rs_table[i].line = serial_req->line;
> - rs_table[i].membase = serial_req->membase;
> - rs_table[i].regshift = serial_req->regshift;
> + return 0;
> }
>
> struct kgdb_serial kgdb8250_serial_driver = {
> - .read_char = kgdb8250_read_char,
> - .write_char = kgdb8250_write_char,
> - .hook = kgdb8250_hook
> + .read_char = kgdb_get_debug_char,
> + .write_char = kgdb_put_debug_char,
> + .hook = kgdb_hook_io,
> };
>
> +void
> +kgdb8250_add_port(int i, struct uart_port *serial_req)
> +{
> + rs_table[i].io_type = serial_req->iotype;
> + rs_table[i].port = serial_req->line;
> + rs_table[i].irq = serial_req->irq;
> + rs_table[i].iomem_base = serial_req->membase;
> + rs_table[i].iomem_reg_shift = serial_req->regshift;
> +
> + /* We will want to look in the rs_table now. */
> + serial_from_rs_table = 1;
> +}
> +
> +/*
> + * Syntax for this cmdline option is
> + * kgdb8250=ttyno,baudrate
> + */
>
> -static int __init kgdb8250_opt(char *str)
> +static int __init
> +kgdb8250_opt(char *str)
> {
> if (*str < '0' || *str > '3')
> goto errout;
> @@ -382,7 +492,12 @@

```

Linux-Kernel: Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

```
> kgdb8250_baud != 38400 && kgdb8250_baud != 57600 &&
> kgdb8250_baud != 115200)
> goto errout;
> +
> + /* Make the baud rate change happen. */
> + gdb_async_info.state->custom_divisor = BASE_BAUD / kgdb8250_baud;
> +
> kgdb_serial = &kgdb8250_serial_driver;
> +
> return 1;
>
> errout:
> diff -Nru a/include/linux/kgdb.h b/include/linux/kgdb.h
> --- a/include/linux/kgdb.h Wed Feb 25 14:21:26 2004
> +++ b/include/linux/kgdb.h Wed Feb 25 14:21:26 2004
> @@ -25,6 +25,8 @@
> extern atomic_t kgdb_killed_or_detached;
> extern atomic_t kgdb_might_be_resumed;
>
> +extern volatile int kgdb_connected;
> +
> extern struct task_struct *kgdb_usethread, *kgdb_contthread;
>
> enum kgdb_bptype {
> diff -Nru a/kernel/kgdb.c b/kernel/kgdb.c
> --- a/kernel/kgdb.c Wed Feb 25 14:21:26 2004
> +++ b/kernel/kgdb.c Wed Feb 25 14:21:26 2004
> @@ -63,7 +63,7 @@
> * has connected to kgdb.
> */
> int kgdb_initialized = 0;
> -static volatile int kgdb_connected;
> +volatile int kgdb_connected;
>
> /* If non-zero, wait for a gdb connection when kgdb_entry is called */
> int kgdb_enter = 0;
> @@ -214,6 +214,7 @@
> do {
> /* wait around for the start character, ignore all other characters */
> while ((ch = (kgdb_serial->read_char() & 0x7f)) != '$') ;
> + kgdb_connected = 1;
> checksum = 0;
> xmitcsum = -1;
>
```

--

George Anzinger george@mvista.com

High-res-timers: <http://sourceforge.net/projects/high-res-timers/>

Preemption patch: <http://www.kernel.org/pub/linux/kernel/people/rml>

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

Linux-Kernel: Re: [Kgdb-bugreport] [PATCH][1/3] Update CVS KGDB's serial driver

More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>